# 文档目标

评估基于BINLOG的CDC对MYSQL数据库的影响，主要包括CPU、内存、磁盘IO。

# 评估方法

## 评估概要

为快速完成评估目标，使用FLINK CDC CONNECTOR（DEBEZIUM）进行CDC，无需部署KAFKA CONNECTOR服务/DEBEZIUM插件。

评估前，使数据库静止，获取当前服务器性能快照。

评估时，运行CDC程序，同时运行SYSBENCH基准测试程序，并持续收集服务器性能快照；停止CDC程序，同时运行SYSBENCH基准测试程序，并持续收集服务器性能快照。

对比前后性能快照、量化CDC程序的影响程度。

# 环境描述

|  |  |  |  |
| --- | --- | --- | --- |
| 物理IP | 硬件 | 软件 | 描述 |
| 88.88.16.113 | 4C 16G | MYSQL 5.7  NODE\_EXPORTER  MYSQLD\_EXPORTER | MYSQL为数据库服务软件  NODE\_EXPORETER采集主机信息  MYSQLD\_EXPORTER采集MYSQL信息 |
| 88.88.16.112 | 4C 16G | SYSBENCH | MYSQL测试工具 |
| 88.88.10.10 | 8C 16G | RPOMETHEUS  GRAFANA | RPOMETHEUS收集采集信息  GRAFANA图形化展示采集信息 |

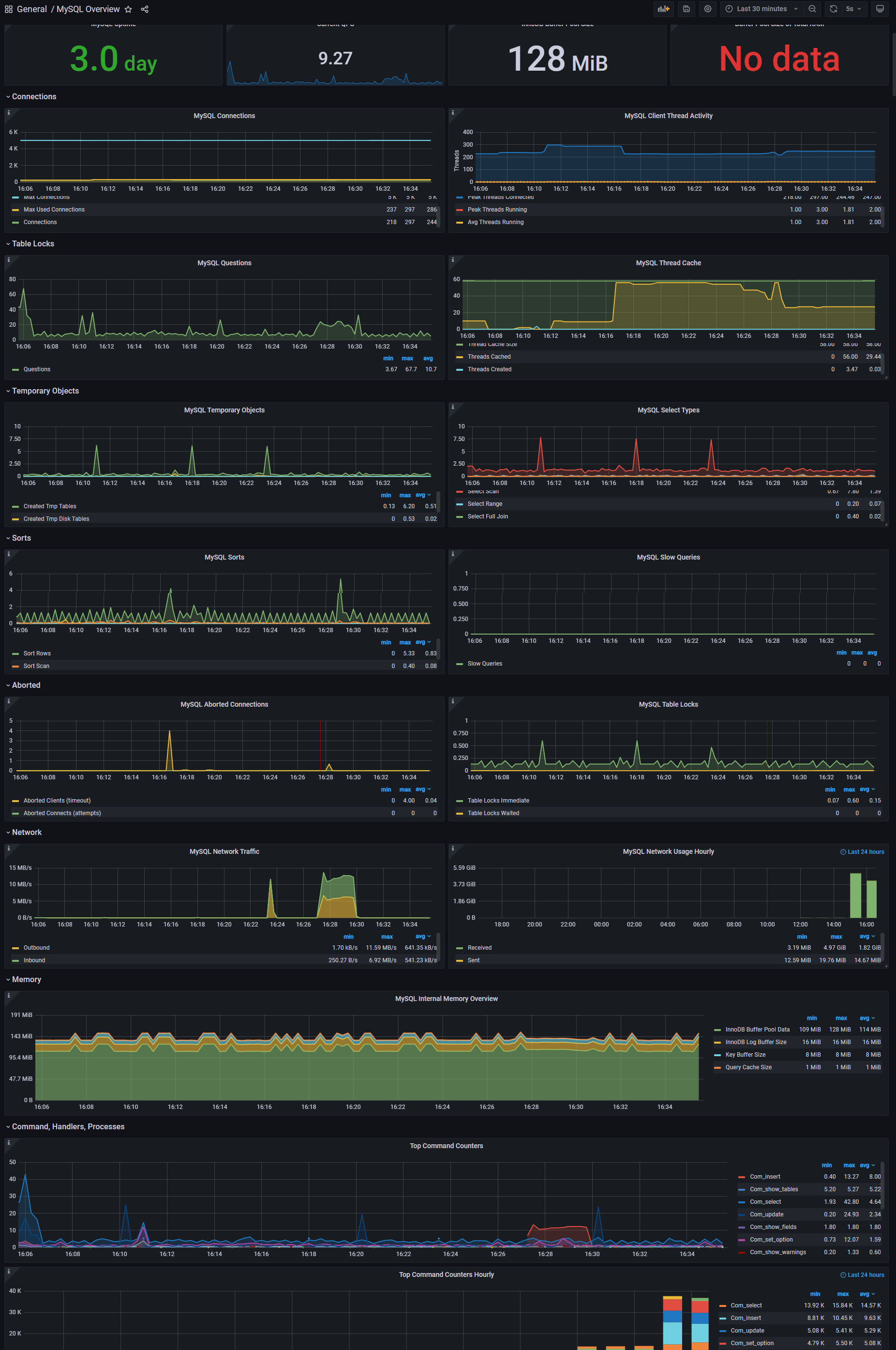
# 评估步骤

## 记录当前系统快照

16:31将用于模拟CDC的测试数据（500W）入库完成，并记录当前快照。



入库完成后，系统基本处于闲置状态。



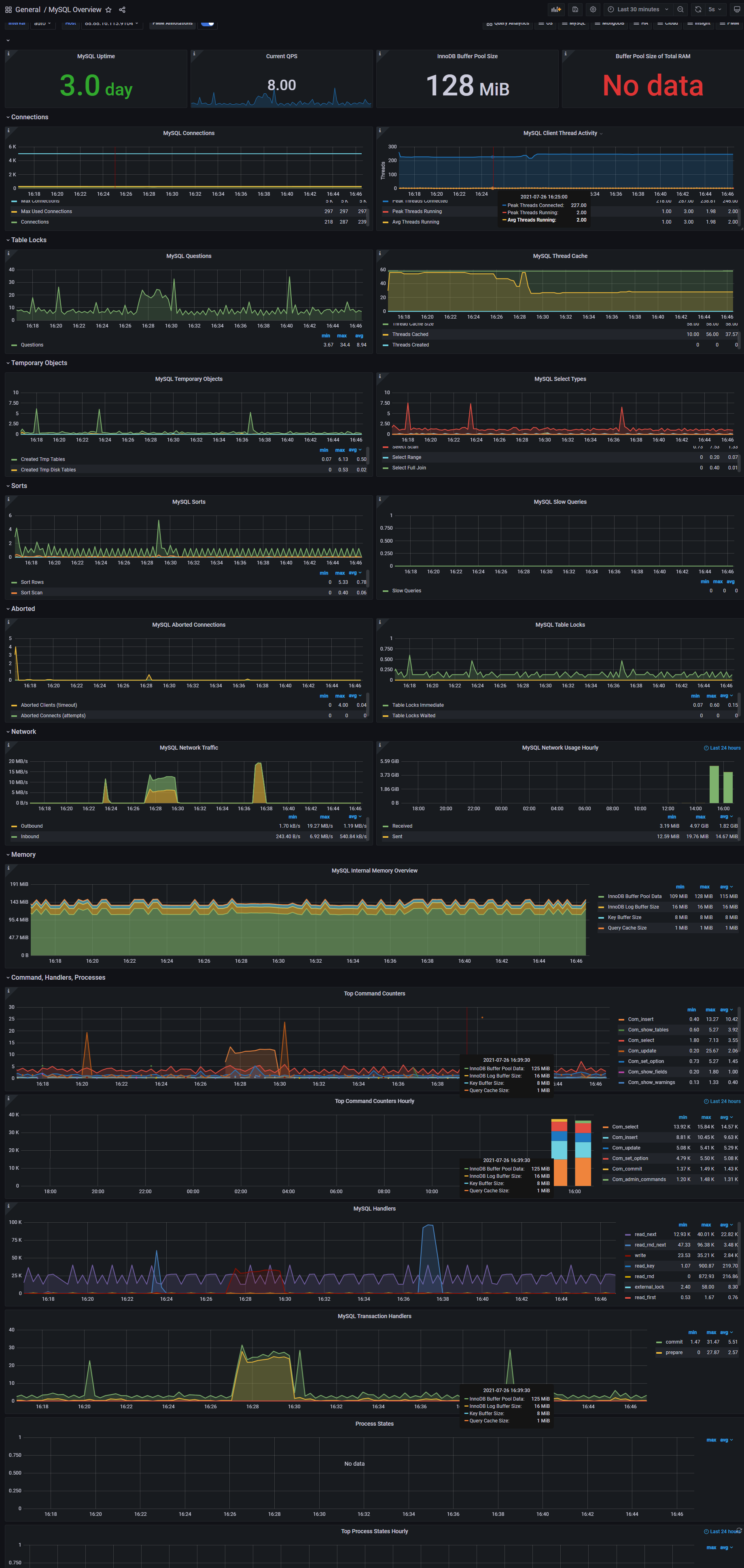
MYSQL基本处于闲置状态。

## 执行CDC动作

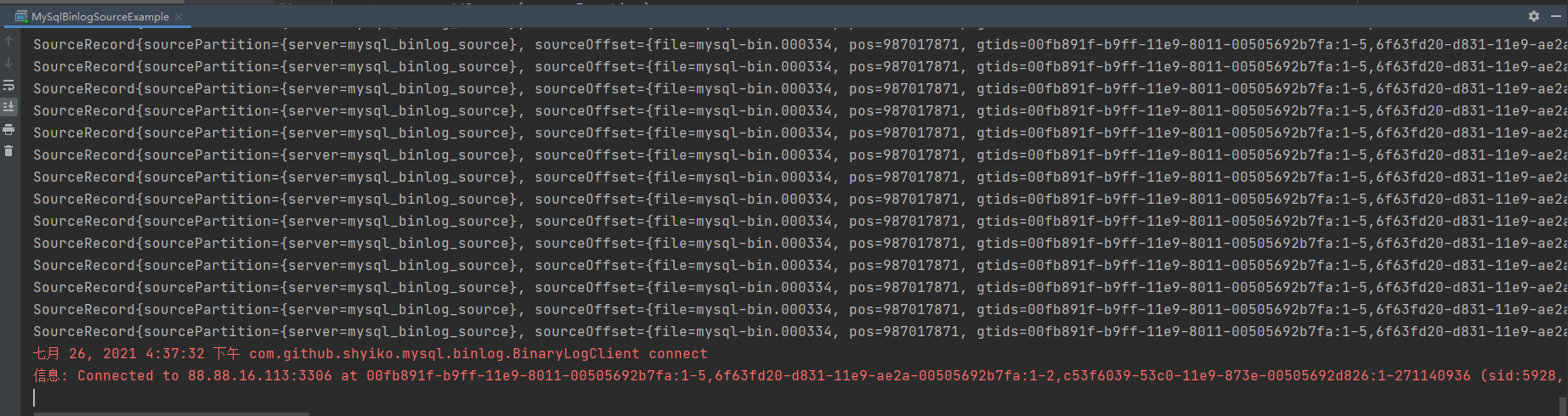
2021-7-26 16:36:40执行CDC程序，于2021 4:37:32同步完存量数据，已开始进行日志的CDC。



从CPU的情况来看，在进行全量快照读取时，占据CPU资源5%不到，内存没有发生抖动。全量初始化完成后，因无事务操作，CDC几乎不占用资源。



CDC期间，数据库状态无异样。



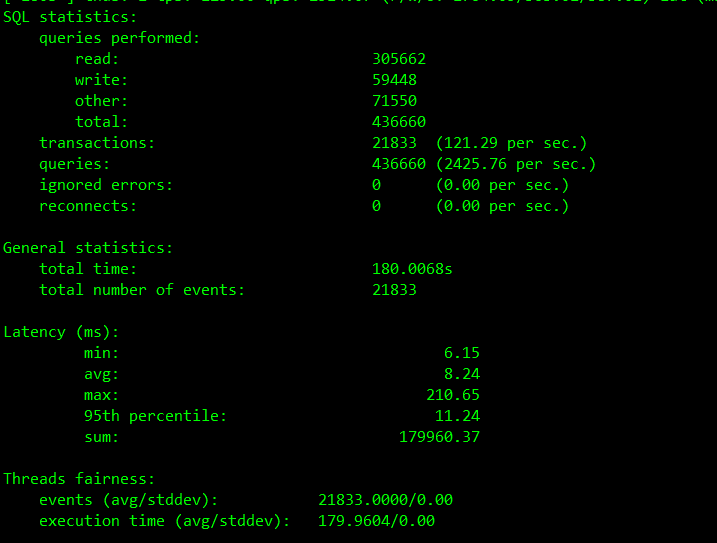
CDC程序初始化存量数据完成，等待解析新增日志。

## 进行压力测试

使用SYSBENCH进行压力基准测试，记录结果。

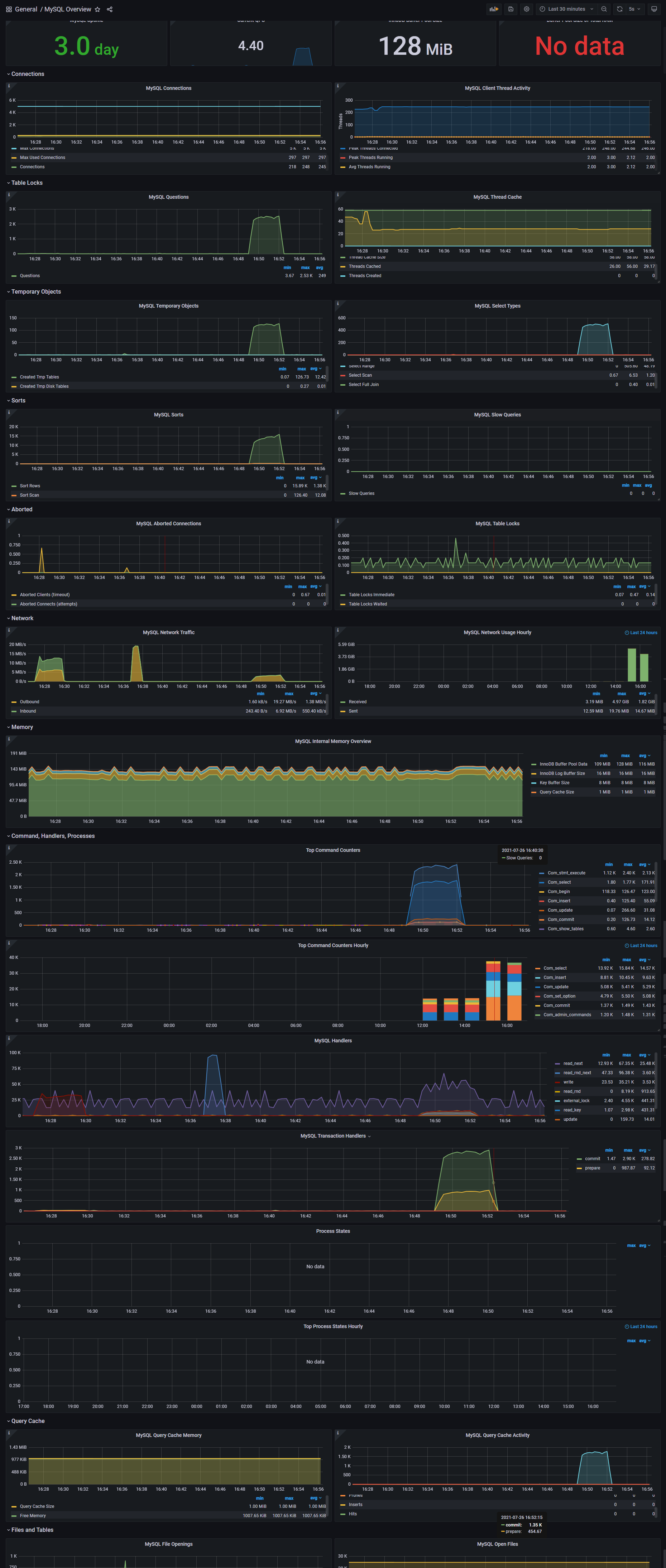
2021-7-26 16:49:10进行SYSBENCH压力测试，

2021-7-26 16:49:10 + 180S执行完毕，报告结果如下：



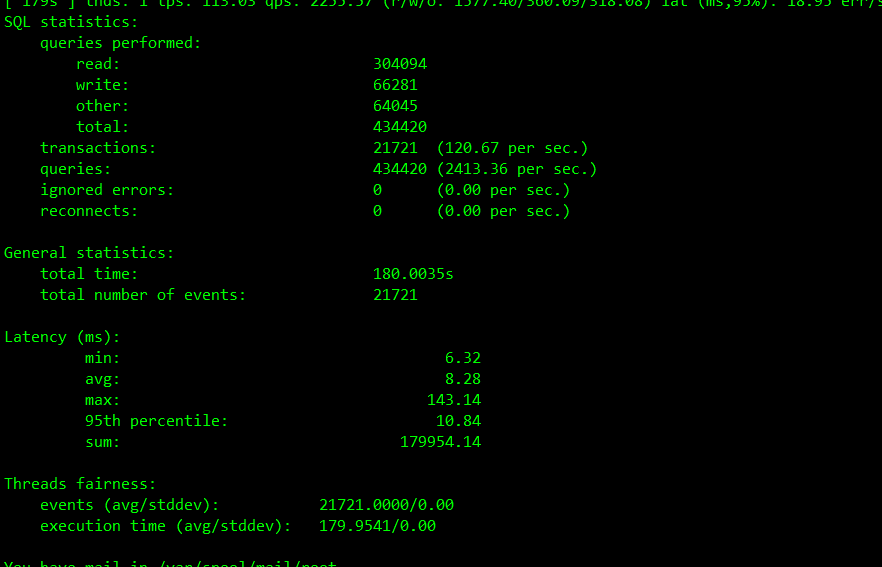


系统CPU飙升明显，且IO发生较为严重的等待。

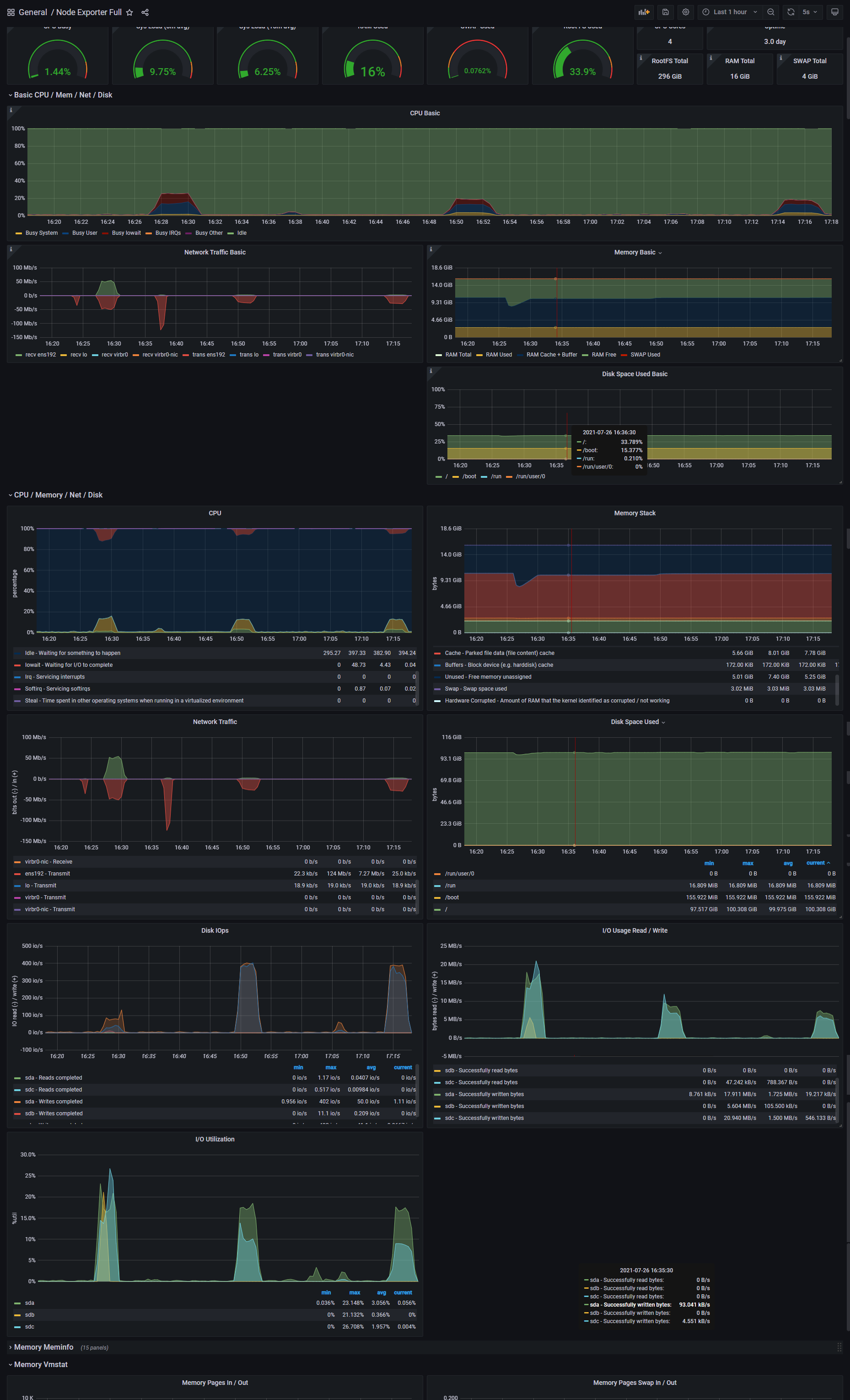


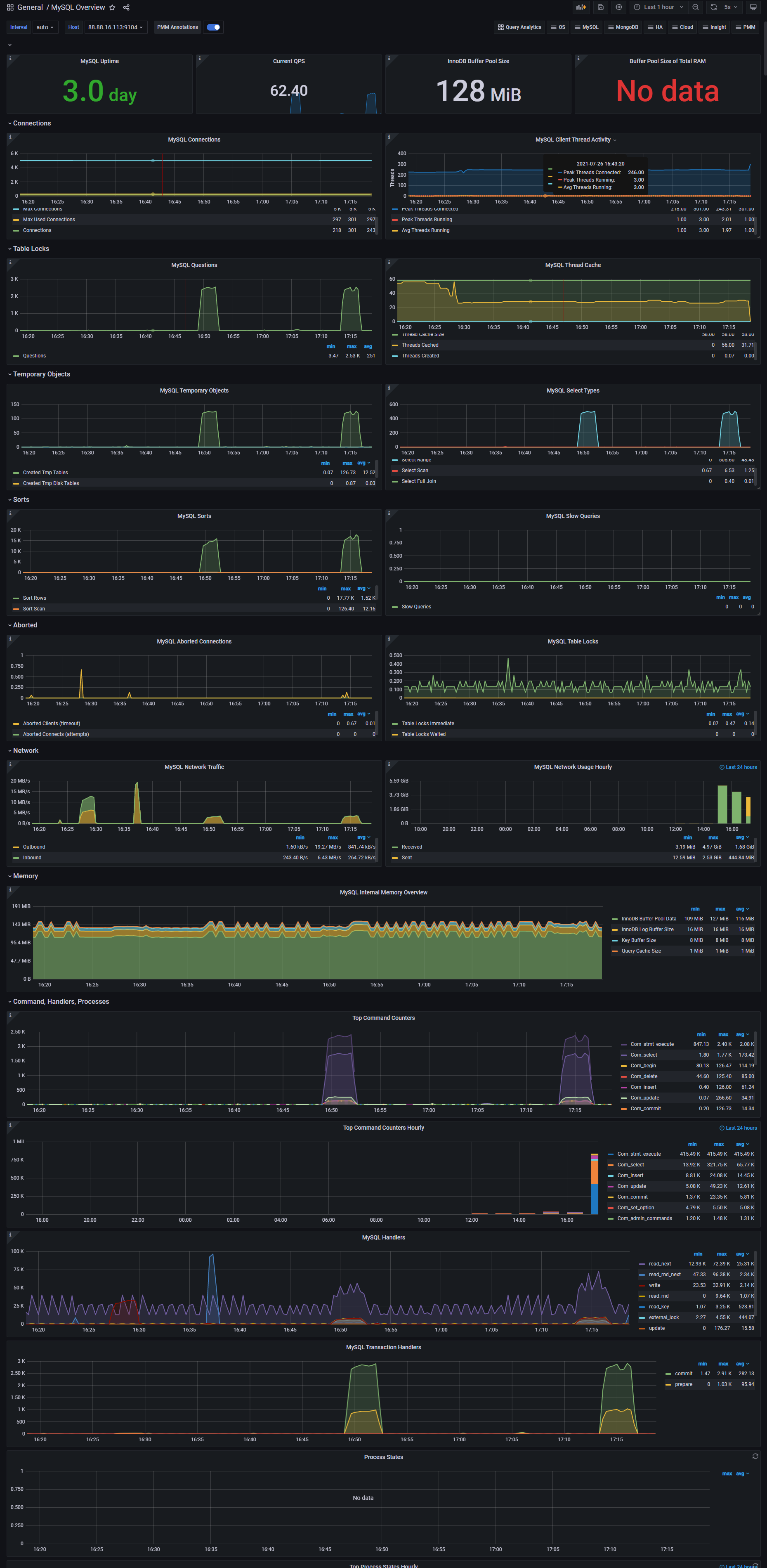
## 再次进行压力测试

停止CDC程序，再次进行压力测试，排除CDC对数据库的影响。

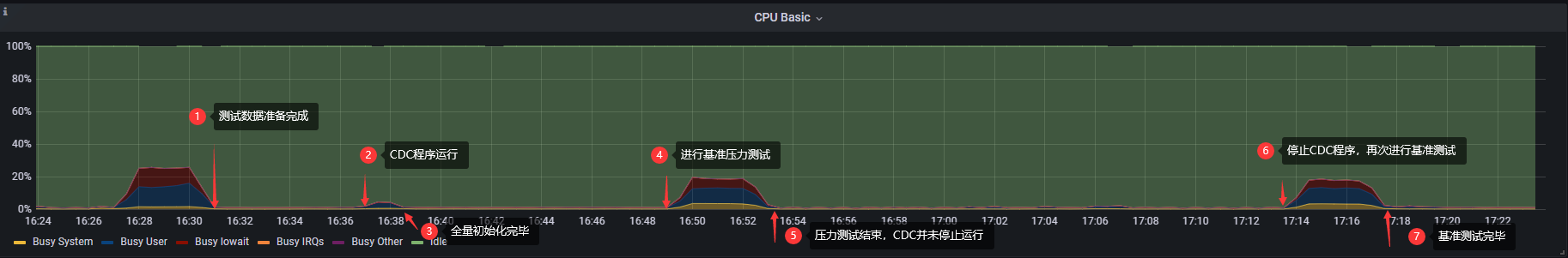


可见，同样的基准测试，有无CDC程序运行，差异并不大。





# 综合分析



从2到3步骤，CDC全量初始化时对数据库形成肉眼可见的压力，在初始化完毕后，压力趋于水平，几乎可以忽略。

在4进行基准测试时，数据库的压力明显飙升，因为仅有一张表进行基准测试，压力并未顶到50%以上，同步观察CDC程序，正常解析数据。

在6中停止CDC程序，排除其对数据库造成的影响，再次进行基准测试，其结果与4步骤的基准测试结果进行对比，发现差异并不大。

# 结论

基于CDC进行增量解析MYSQL日志的性能损耗，量化公式：

（4步骤中系统指标值 - 6步骤中系统指标值）/ 系统指标值

(323.44-316.48)/393.45=0.01768，不到2%的性能损耗。

# 问题

由于当前采用的CDC技术，默认会进行全量快照初始化，对数据库造成的压力随同步库表数量增多成正相关，如果进行全库初始化将是灾难，或面临快照过久问题，直接导致CDC程序崩溃。

解决：在低峰期或停止应用前提下，进行初始化；全量初始化，由另起程序进行柔和控制，CDC仅做增量同步。

# 附录

CDC代码

public class MySqlBinlogSourceExample {  
 public static void main(String[] args) throws Exception {  
 SourceFunction<String> sourceFunction = MySQLSource.<String>*builder*()  
 .hostname("88.88.16.113")  
 .port(3306)  
 // databaseList exclusive with tableList  
 // monitor all tables under test database  
// .databaseList("test")  
 .tableList("test.sbtest1")  
 .username("root")  
 .password("root")  
 // converts SourceRecord to String  
 .deserializer(new StringDebeziumDeserializationSchema())  
 .build();  
  
 StreamExecutionEnvironment env = StreamExecutionEnvironment.*getExecutionEnvironment*();  
  
 env.addSource(sourceFunction)  
 // use parallelism 1 for sink to keep message ordering  
 .print().setParallelism(1);  
  
 env.execute();  
 }  
}